Departamento de Ingeniería Electrónica, de Sistemas Informáticos y Automática

INFORMÁTICA INDUSTRIAL II

MANUAL DE USO DEL MONITOR PAULMON

CURSO 2001-2002

ÍNDICE

PaulMon fue creado por Paul Stoffregen.

PAULMON2 Documentation, © Paul Stoffregen
http://www.pjrc.com/tech/8051/pm2_docs/problems.html
Last updated: October 17, 1999

1. Introducción.

Pasamos a describir el programa monitor empleado en las prácticas de la asignatura. PAULMON2 es un programa monitor fácil de usar, indicado para su uso en una placa computadora simple basada en un microcontrolador compatible con el 8051.

Colocando PAULMON2 en la EPROM de arranque de la placa, esta "arrancara" con un monitor de línea de comandos (como por ejemplo el MS-DOS), que podemos controlar usando un programa corriente de emulación de terminal (como el HyperTerminal de Windows). PAULMON2 nos permite descargar los programas a RAM y ejecutarlos, lo cual es mucho más rápido y fácil que reprogramar la EPROM. PAULMON también proporciona varias herramientas para ayudar a depurar nuestros programas.

PAULMON2 proporciona también varias funciones internas que podemos llamar desde nuestros programas. Algunas de estas son bastante útiles para depurar nuestro código.

Puesto que PAULMON2 es un programa gratuito, cuando hayamos terminado de diseñar nuestro programa, cualquier trozo de PAULMON2 que hayamos usado en nuestro programa puede copiarse directamente en el código para hacer el programa una aplicación autocontenida.

PAULMON2 también puede usar una ROM Flash. Las placas con Flash ROM se pueden hacer "dedicadas" al descargar una copia ligeramente modificada del programa de aplicación, y PAULMON2 ejecutará este programa cuando se resetee la placa en lugar de la pantalla de bienvenida habitual. Si el código debe actualizarse después, se puede colocar un jumper en el diseño hardware que provoque que PAULMON2 borre la Flash ROM y vuelva a modo de desarrollo, permitiendo de esta forma que se pueda descargar una nueva versión del programa.

2. Necesidades de Hardware.

Se puede trabajar sobre dos tipos de plataformas.

Sistema Mínimo.

El sistema mínimo necesario para ejecutar PAULMON2 se muestra en la figura 1. El procesador 8752 o equivalente debe ser programado con una copia de PAULMON2. El único chip adicional necesario es un MAX232, para convertir las señales +5v del 8051 a los niveles que usa la interfaz RS-232. Por supuesto, es necesario una fuente de alimentación de +5v y un cable serie estándar.

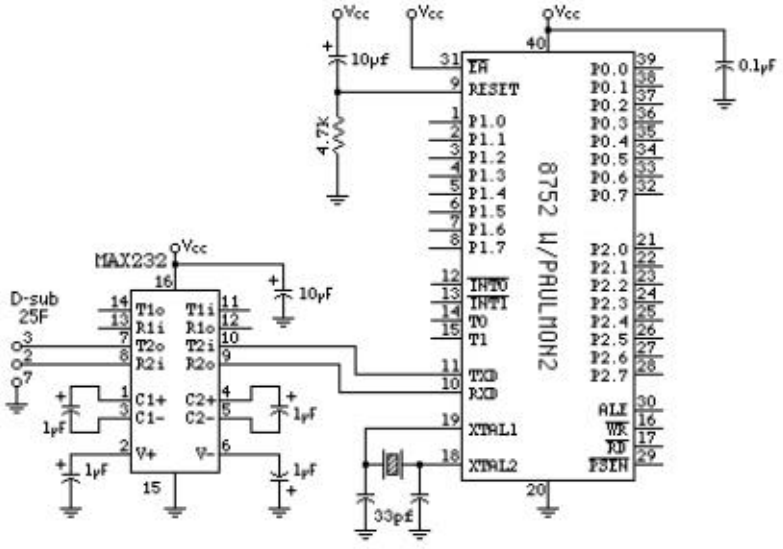

Figura 1: Sistema mínimo necesario para ejecutar PAULMON2

Este sistema no es muy útil…. no podemos descargar programas porque no hay chip de RAM externa, pero este sistema "arrancará" y PAULMON2 se ejecutará correctamente. Si está disponible un chip programable de la familia 8051 (8751, 87C51, 87C52, 89C51, 89C52, etc), y se emplea una técnica de construcción de prototipos (wire wrap, cable

punto a punto, etc), **es más fácil construir y probar este sistema mínimo primero**, y luego añadir la RAM externa y otros componentes cuando el sistema mínimo esté funcionando.

Por supuesto, el chip 89C52 debe programarse con PAULMON2, por lo que se necesita primero un programador que pueda escribir el 89C52 inicialmente para programarlo.

Sistema Útil.

Para construir un sistema útil, al menos hay que disponer de un chip de RAM externo de forma que el código (además del de PAULEMON2 en sí) pueda ejecutarse. Al menos se necesitan dos chips más, como se muestra aquí:

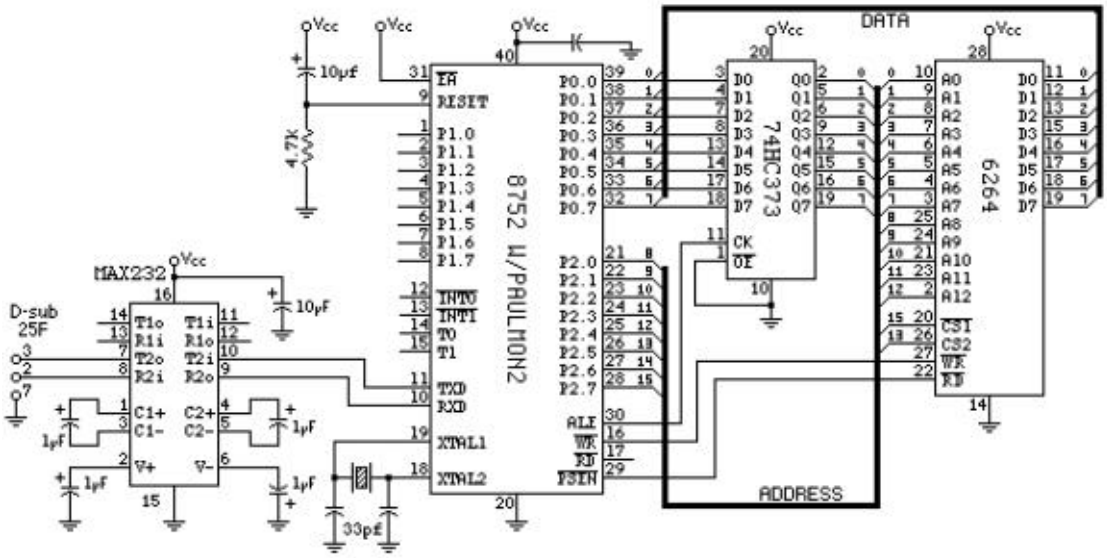

Figura 2: Sistema útil mínimo para PAULMON2

Puesto que la señal RD del 8051 (patilla #17) no se emplea, la instrucción MOVX no puede emplearse para leer de la RAM, si no que habrá que emplear MOVC. Sin embargo, al usar la línea PSEN en lugar de la RD nos permite ejecutar código desde la RAM, que es importante para desarrollar código con PAULMON. En algunos sistemas, las líneas RD y PSEN se combinan mediante una puerta AND (74HC08) de tal forma que ambas instrucciones MOVX y MOVC pueden leer la RAM.

Este esquema conecta simplemente las dos patillas de selección de chip a las líneas A13 y A15, lo que provoca que los 8k de memoria aparezcan en dos direcciones, 0x2000 a 0x3FFF, y 0x6000 a 0x7FFF.

Puesto que no se dispone de ningún hardware adicional, los programas escritos para ejecutarse en esta placa solo podrán realizar E/S mediante el puerto serie. La mayoría de las aplicaciones necesitan hardware adicional de E/S, pero este sistema mínimo puede emplearse para probar programas simples que se comuniquen por el puerto serie.

Procesadores Compatibles

Casi todos los procesadores compatibles con el 8051 lo serán con PAULMON2. Algunos procesadores 8051 contienen hardware especial, el cual PAULMON2 no usará, pero este podrá usarse por los programas que se descarguen empleando el monitor.

Interfaz de Puerto Serie

El esquema anterior muestra el chip MAX232 chip haciendo de interfaz entre el 8051 y el puerto serie del PC. Se pueden emplear otros chips como los 1488/1489 , pero para una placa prototipo con solo 5 voltios, el MAX232 (y chips equivalentes) pueden generar los voltajes +/- 12V necesarios para realizar la

interfaz, de esta forma es más dificil cometer errores de cableado que puedan destruir los componentes de la placa si solo se emplea una fuente de alimentación de 5 V.

Mapeo de Memoria y RAM

PAULMON2 no necesita RAM para funcionar, pero los programas no se podrán descargar a no ser que RAM (o Flash ROM) esté presente para guardar los datos descargados. Las versiones estándar de PAULMON2 esperan encontrar RAM en las direcciones 0x2000, 0x4000, o 0x8000. Dentro del código de PAULMON existen instrucciones LJMP que provocan que las instrucciones salten a las direcciones asociadas de RAM (esto es, 0x0003 salta a 0x2003 para la interrupción INT0). Es posible configurar PAULMON2 para usar RAM localizada en otras direcciones.

Para poderse emplear, la RAM debe conectarse de forma que se pueda leer como **memoria de código**. En otras palabras, la línea PSEN debe ser capaz de seleccionar una lectura de la RAM.

Flash ROM

PAULMON2 puede dar soporte a Flash ROM. PAULMON2 fue diseñado y probado con el chip de Flash ROM de AMD 28F256. Pueden emplearse chips de más capacidad, aunque contengan más memoria del espacio de direcciones disponible en el 8051 de 64K, por lo que solo una porción podrá usarse. AMD ofrece un chip nuevo 28F256A, que dispone de una interfaz de programación más simple. PAULMON2 no ha sido probado con este chip. Los chips de la serie 29Fxxx todavia no se pueden programar con PAULMON2.

3.- Requerimientos Software.

Se necesitan dos programas en nuestro PC para usar PAULMON.

Emulador de Terminal

Se hace necesario un programa emulador de terminal para comunicar con nuestro hardware basado en 8051 que esté ejecutando PAULMON. Aunque casi cualquier emulador de terminal puede funcionar, los siguientes programas se han probado y funcionan:

- ProComm (MSDOS)
- Seyon (X11, Linux)
- Terminal (Windows 3.1)
- HyperTerminal (Windows 95) - nota: la nueva configuración puede no tener efecto hasta que se sale del programa y se vuelve a entrar.

El formato de datos es 8 bits, sin paridad, 1 bit de stop, **sin control de flujo**. Si tenemos una copia de PAULMON configurada para una velocidad de transmisión y cristal particular, tendremos que usar esta tasa de transmisión en el emulador de terminal. En otro caso, PAULMON intentará detectar la configuración de velocidad de transmisión del programa emulador de terminal en el arranque cuando pulsemos Enter (se espera el código ASCII 13). Por supuesto, solo las velocidades de transferencia que pueda producir el 8051 con nuestro cristal de cuarzo serán detectadas….. es mejor probar una tasa de transmisión lenta primero (300 o 1200) y luego probar otras más rápidas cada vez que reiniciemos (alimentación desconectada, no solo reset).

Algunos programa emuladores de terminal pueden enviar datos cuando son arrancados, lo que puede confundir la detección automática de velocidad en PAULMON. Es mejor conectar la placa 8051 en el PC (con la alimentación desconectada), ejecutar el programa de emulación de terminal, y luego aplicar la alimentación a la placa y pulsar Enter.

Casi todas las prestaciones de PAULMON realmente no necesitan emulación de ninguna terminal en particular. La excepción es el editor de memoria en el paquete de extras, necesita emulación de terminal VT100 para funcionar. Si la emulación VT100 no funciona, las otras características no se verán afectadas.

Ensamblador

Un ensamblador no es estrictamente necesario para usar PAULMON, pero sin el ensamblador es muy difícil escribir código que pueda descargarse al hardware basado en el 8051 vía el comando de Descarga.

Casi cualquier ensamblador del 8051 que produzca una salida en formato Intel-Hex debe funcionar, aunque algunos ensambladores del 8051 puedan tener una sintaxis de entrada ligeramente diferente. PAULMON2 ha sido escrito usando en ensamblador AS31, que está disponible de forma gratuita.

4.- Puesta en marcha.

Después de haber fabricado el hardware necesario (incluyendo la programación de PAULMON2 en la memoria EPROM), e instalado el programa de emulación de terminal, estaremos listos para comenzar.

1. Conectar el hardware del 8051 (alimenacion apagada) al puerto serie del PC.
2. Arrancar el programa de emulación de terminal en el PC.
3. Verificar la configuración del programa (puerto, velocidad, 8-N-1, sin control de flujo).
4. Aplicar tensión al hardware 8051.
5. Pulsar Enter (para que PAULMON2 pueda determinar la velocidad de transmisión)

Si se ha configurado la detección automática de velocidad (lo configurado por defecto), PAULMON2 esperará a que pulsemos Enter (Retorno) tras dar tensión a la placa del 8051. Emplea este carácter entrante para determinar la tasa de transmisión (baud rate). El resultado se almacena en memoria de forma que la detección de tasa de transmisión no sea necesario repetirla a no ser que la memoria sea borrada o sobreescrita. Una vez que se conoce la tasa de transmisión correcta, PAULMON2 responderá con un mensaje:

```
                Welcome to PAULMON2 (beta4), by Paul Stoffregen

  See PAULMON2.DOC, PAULMON2.EQU and PAULMON2.HDR for more
information.
```

Si nuestra copia de PAULMON2 tiene alguno de los comandos adicionales instalados, apareceran listados junto al mensaje inicial, que viene seguido de la petición de comandos de PAULMON2:

```
PAULMON2 (beta4) Loc:2000 >
```

El "2000" indica el valor del puntero de memoria de PAULMON2, que se emplea en varios comandos que permiten manipular la memoria de la placa. Pulsando '?' mostrará un breve mensaje de ayuda, que proporciona una lista de comandos disponibles.

Es una buena idea es probar la memoria RAM empleando el editor de memoria mejorado o el simple editor si se ha seleccionado una versión de PAULMON2 de 4K de tamaño. Una vez que podamos escribir en la RAM, podemos probar uno de los programas de ejemplo, y estaremos seguros de que podemos escribir nuestro propio código.

5.- Comandos estándar.

PAULMON2 dispone de nueve comandos estándar. Las teclas asignadas a cada comando pueden configurarse cuando se ensambla PAULMON2. Las letras mostradas aquí representan las teclas por defecto de cada comando

- **M** – Listar programas
Intenta encontrar programas en memoria y muestra una lista. Para que un programa se pueda encontrar, debe tener una cabecera de programa de 64 bytes. Por ejemplo:

```
PAULMON2 (beta4) Loc:0100 > M

Program Name                    Location      Type
  List                            1000      External command
  Single-Step                     1400      External command
  On-Line Docs                    1700      External command
```

R – Ejecutar programa
Nos permite ejecutar programas que tengan una cabecera de 64 bytes indicado que son programas. Esto nos proporciona una manera simple de ejecutar un programa sin tener que especificar a que dirección de memoria saltar. Para ejecutar programas sin la cabecera de 64 bytes, usaremos el comando Jump. Mientras que se esté ejecutando el programa, este tiene control completo sobre el procesador... PAULMON2 ya no se está ejecutando. Si el programa falla (bucle infinito), la única forma de retornar al monitor PAULMON2 es hacer un reset al procesador.

D - Descarga
El comando de descarga permite que PAULMON2 reciba un fichero en formato Intel-Hex, que se escribe en la memoria a medida que se va recibiendo. Tras ejecutar este comando, necesitamos transmitir nuestro fichero intel-hex a PAULMON2. Puesto que el formato Intel-Hex es un formato texto, la opción de envío ascii del programa de emulación de terminal debe funcionar. No se debe emplear un protocolo como xmodem o zmodem. El formato Intel-Hex incluye información específica de donde se deben escribir los datos en memoria, por lo que el valor del puntero de memoria no se emplea con este comando.

Cuando se descarga a la RAM, PAULMON2 puede aceptar datos a la máxima velocidad posible, por lo que no se hace necesario usar pausa entre caracteres u otros retardos. Para abortar una descarga, solo pulsar enter. A medida que PAULMON2 recive cada línea de datos Intel-Hex, imprime un punto. Cuando se finaliza una descarga o se aborta, se muestra un breve resumen.

```
PAULMON2 (beta4) Loc:2000 > Download

Begin ascii transfer of Intel hex file, or ESC to abort

.....................................................................
.....................................................................
.....................................................................
.....................
Download completed

Summary:
 249  lines received
 3923  bytes received
 3923  bytes written
No errors detected
```

Si el comando de descarga informa de errores intentando escribir, como:

```
Summary:
 246  lines received
 3834  bytes received
 630  bytes written
```

```
Errors:
 3204  bytes unable to write
```

esto significa que ha recibido correctamente el fichero Intel-Hex, pero la memoria especificada por el fichero Intel-Hex es de solo lectura. En este ejemplo, varias direcciones de solo lectura contenían el mismo dato (porque era una nueva versión del mismo código), por lo que PAULMON2 verificó 630 de los bytes como correctamente escritos.

Cuando PAULMON2 recibe una entrada incorrecta, se imprime un resumen similar a este:

```
Summary:
 3  lines received
 62  bytes received
 62  bytes written
Errors:
 1  bad checksums
 2  unexpected begin of line
 17  unexpected hex digits
 2  unexpected non hex digits
```

U - Transferencia
El comando de transferencia nos permite especificar una porción de la memoria que va a ser transmitida al PC en formato Intel-Hex. Después de especificar el rango de memoria, PAULMON2 espera la pulsación de una tecla antes de transmitir, lo que nos permite activar un comando de "captura" en nuestro programa de emulación de terminal para guardar los datos entrantes a disco. Una transferencia típica puede ser como esto:

```
PAULMON2 (beta4) Loc:0100 > Upload

First Location: 0000
Last Location:  003A

Sending Intel hex file from  0000  to  003A
Press any key:
:100000000208A60220037420116A2202200B742D1C
:10001000116A22022013020A72FFFF02201BFFFF57
:10002000FFFFFF0220230209AAFFFF02202B216EFF
:0B003000016A0162215D217F218C012B
:00000001FF
```

N – Nueva dirección
Esto nos permite cambiar el puntero de memoria:

```
PAULMON2 (beta6) Loc:8000 > New location

New memory location: 2C00

PAULMON2 (beta6) Loc:2C00 >
```

J – Saltar a dirección de memoria
El comando de salto nos permite saltar directamente a un programa. PAULMON2 introduce 0000 en la pila, de forma que un programa que termine con la instrucción "RET" vuelva a ejecutar PAULMON2. Si el programa falla (bucle infinito), la única forma de volver a PAULMON2 es resetear el procesador.

H – Volcado Hexadecimal de la memoria
El comando de volcado nos permite inspeccionar la memoria externa. Se imprimen 256 bytes en formato hexadecimal y ascii, comenzando en la dirección del puntero de memoria. Los datos son leidos de la memoria de programa (con MOVC).

```
PAULMON2 (beta6) Loc:2100 > Hex dump memory
```

```
2100: FD 31 3F 40 C6 CD 12 20 6A ED C0 E0 31 1D D0 E0   }1?@FM  jm@`1 P`
2110: 2A FA E4 3B FB DC B4 C3 8A 82 8B 83 22 EB C4 54   *zd;{ \4C    "kDT
2120: F0 FB EA C4 54 0F 4B FB EA C4 54 F0 FA 22 EA C4   p{jDT K{jDTpz"jD
2130: 54 0F FA EB C4 54 F0 4A FA EB C4 54 0F FB 22 C3   T zkDTpJzkDT {"C
2140: C0 F0 94 30 F5 F0 94 0A 40 06 E5 F0 94 07 F5 F0   @p 0up  @ ep  up
2150: E5 F0 C3 54 F0 60 01 D3 E5 F0 D0 F0 22 C0 E0 C4   epCTp` SepPp"@`D
2160: 54 0F 24 F6 50 02 24 07 24 3A 11 6A D0 E0 C0 E0   T $vP $ $: jP`@`
2170: 54 0F 24 F6 50 02 24 07 24 3A 11 6A D0 E0 22 C0   T $vP $ $: jP`"@
2180: E0 E5 83 31 5D E5 82 31 5D D0 E0 22 C0 E0 E4 93   `e 1]e 1]P`"@`d
2190: A3 60 0A A2 E7 54 7F 11 6A 40 02 80 F1 D0 E0 22   #` "gT  j@  qP`"
21A0: 90 2D 50 11 45 EF 31 5D EE 31 5D 11 06 75 83 10     -P Eo1]n1]  u
21B0: F1 BB 50 2F 75 82 08 E4 93 B4 A5 23 05 82 E4 93   q;P/u  d 4%#  d
21C0: B4 64 1C C0 83 74 DA C0 E0 74 21 C0 E0 05 82 E4   4d @ tZ@`t!@`  d
21D0: 93 C0 E0 05 82 E4 93 C0 E0 22 D0 83 12 20 06 05    @`  d @`"P
21E0: 83 21 B0 90 2D 60 31 8C 11 62 11 16 75 F0 A0 C0    !0 -`1  b  up @
21F0: F0 75 F0 21 C0 F0 F5 F0 90 10 00 F1 BB 50 21 75   pup!@pup   q;P!u
```

I – Volcado Hexadecimal de la memoria interna

Este comando imprime en pantalla un volcado de la memoría interna, por ejemplo:

```
PAULMON2 (beta7) Loc:FFF0 > Hex dump internal memory

00: 00 00 00 00 43 03 F0 FF 55 55 55 55 AA AA AA AA
10: 55 55 55 55 AA AA AA AA 55 55 55 55 AA AA AA AA
20: 55 55 55 55 AA AA AA AA 55 55 55 55 AA AA AA AA
30: 55 A0 01 E6 07 0A 00 01 43 31 0F 48 0B 00 0B AA
40: 55 55 55 55 AA AA AA AA 55 55 55 55 AA AA AA AA
50: 55 55 55 55 AA AA AA AA 55 55 55 55 AA AA AA AA
60: 55 55 55 55 AA AA AA AA 55 55 55 55 AA AA AA AA
70: 55 55 55 55 AA AA AA AA E2 33 AA FF AA AA AA AA
```

Esta función puede ser útil para examinar la memoria interna tras la ejecución de un programa. Sin embargo, PAULMON2 usa RAM interna, por lo que los siguientes rangos de memoria probablemente sean sobreescritas por PAULMON2, mostrados en rojo arriba.

Rango de Memoria	Uso por PAULMON2
00 a 07	Registros R0 a R7
10 a 1F	Usados durante la Descarga (cuentas de Error)
31 a 40	Pila
78 a 7B	Almacena la velocidad para arranque en caliente

E – Edición de ram externa
El comando de edición nos permite escribir directamente en la ram externa.

C – Borrar memoria
El comando de borrado nos proporciona una forma fácil de rellenar un rango de memoria con ceros.. Debido a que los chips de RAM contienen “basura” cuando se alimentan, normalmente se hace necesario tener una memoria “limpia”, para que sea fácil ver los datos “reales”.

6.- Comandos extendidos.

Paquete de Extras

Las versiones de PAULMON2 diseñadas para emplearse en al menos 8k de EPROM (esto es, el chip 89C52) contienen un paquete de "extras" que incluye tres comandos adicionales. Las versiones que caben en solo 4k de EPROM (esto es el chip 89C51) no contienen estos comandos extra.

L - Listar
Este comando desensambla datos de la memoria para mostrar el código con nemónicos y operandos. Por ejemplo:

```
PAULMON2 (beta7) Loc:0000 >  List

0000: 02 08 F1      LJMP    08F1
0003: 02 20 03      LJMP    2003
0006: 74 20         MOV     A, #20
0008: 11 6A         ACALL   006A
000A: 22            RET
000B: 02 20 0B      LJMP    200B
000E: 74 2D         MOV     A, #2D
0010: 11 6A         ACALL   006A
0012: 22            RET
0013: 02 20 13      LJMP    2013
0016: 02 0A BD      LJMP    0ABD
0019: FF            MOV     R7, A
001A: FF            MOV     R7, A
001B: 02 20 1B      LJMP    201B
001E: FF            MOV     R7, A
001F: FF            MOV     R7, A
0020: FF            MOV     R7, A
0021: FF            MOV     R7, A
0022: FF            MOV     R7, A
0023: 02 20 23      LJMP    2023
```

Aqui PAULMON2 muestra las instrucciones reales de su propio código. Como esperabamos, en la dirección 0003 (un vector de interrupción), tenemos una instrucción LJMP hacia 2003.

S – Paso a paso
Este comando intentará ejecutar un programa en modo paso a paso. No lo emplearemos en prácticas.

E – Editor ampliado de memoria
Este comando reemplaza al comando simple E disponible en la versión principal de PAULMON2 por una versión mucho más agradable. **La emulación de terminal VT100 es necesaria**, porque este editor de memoria emplea el control de posicionado del cursor. Cuando se ejecuta con un terminal VT100, debe aparecer como esto:

```
CODE        8051 External Memory Editor, Paul Stoffregen, 1996
ADDR: +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F ASCII EQUIVILANT
-------------------------------------------------------------------------
0000: 02 08 F1 02 20 03 74 20 11 6A 22 02 20 0B 74 2D   q   t  j"   t -
0010: 11 6A 22 02 20 13 02 0A BD FF FF 02 20 1B FF FF    j"     =
0020: FF FF FF 02 20 23 02 09 F5 FF FF 02 20 2B 21 6E      #  u    +!n
0030: 01 6A 01 62 21 5D 21 7F 21 8C 01 7F 01 C3 01 26   j b!]! !   C &
0040: 01 16 02 09 76 02 0A E3 01 72 02 0A D0 02 0A 06       v  c r  P
0050: 02 0A 0C 02 0A 4A 02 08 B3 02 08 79 02 08 1B 02        J  3  y
0060: 07 F1 30 98 FD C2 98 E5 99 22 30 99 FD C2 99 F5  q0 }B e "0 }B u
0070: 99 22 C0 E0 74 0D 11 6A 74 0A 11 6A D0 E0 22 C2  "@`t  jt  jP`"B
0080: D5 11 62 11 16 B4 1B 03 D3 E4 22 B4 0D 05 D2 D5 U b  4  Sd"4  RU
0090: C3 E4 22 FA 31 3F 40 E9 CA 11 6A 11 62 11 16 B4 Cd"z1?@iJ j b  4
00A0: 1B 02 80 E4 B4 0D 03 EA C3 22 B4 08 04 11 6A 80     d4  jC"4   j
00B0: D0 B4 15 02 80 F7 FB 31 3F 40 E0 CB 11 6A EA C4 P4   w{1?@`K jjD
00C0: 4B C3 22 7A 00 7B 00 7C 04 C2 D5 11 62 11 16 B4 KC"z { | BU b  4
00D0: 1B 07 D3 E4 F5 83 F5 82 22 B4 08 02 80 03 B4 7F    Sdu u "4    4
00E0: 0C BC 04 02 80 E5 11 6A 31 2E 0C 80 DE B4 0D 10    <   e j1.  ^4
00F0: 8B 83 8A 82 BC 04 07 E4 F5 83 F5 82 D2 D5 C3 22     <  du u RUC"
-------------------------------------------------------------------------
^A=ASCII ^X=Hex ^F=Fill  ^G=Goto  ^C=Code  ^D=Data  ^L=Redraw  ^Q=Quit
```

El editor de memoria dispone de los siguientes comandos:

- CTRL-E - Habilita/Desabilita el modo de edición: Cuando nos encontramos en modo de edición, teclear caracteres ordinarios (sin CTRL) provoca la edición de la memoria, por lo que por defecto está deshabilitado hasta que se pulsa CTRL-E. Cuando la edición está desabilitada, la línea inferior no muestra `^A=ASCII ^X=Hex ^F=Fill` y en su lugar muestra solo `^E-Edit`. La pantalla refleja siempre el contenido actual de la memoría (CODE o DATA), por lo que un intento de escribir memoria de solo lectura no tendrá efecto, incluso si está habilitada la edición.
- CTRL-A – Selección de modo de edición ASCII : En el modo de edición ASCII, cualquier texto tecleado será escrito directamente en memoria comenzando en la posición actual del cursor. La palabra "ASCII" se mostrará resaltada para recordarnos que estamos en modo ASCII de entrada.
- CTRL-X – Selección de modo de edición HEX : En el modo de edición HEX, los datos se introducen como números hexadecimales, que serán escritos en memoria a partir de la posición actual del cursor.. La palabra "HEX" se mostrará resaltada para recordarnos que el modo de entrada HEX está seleccionado.
- CTRL-F – Rellenar un bloque de memoria: Este comando llena un bloque de memoria con un byte particular. Aparecerá un aviso pidiendo las direcciones de memoria inicial y final, y el byte a escribir. Para cancela habrá que pulsar ESC. Por supuesto, este comando solo está disponible si la edición está habilitada.
- CTRL-G – Saltar a una nueva dirección: Este comando nos permite movernos a otras direcciones de memoria. Aparece una petición de la nueva dirección.
- CTRL-C – Mostrar como memoria de código (MOVC) : Fuerza al editor a mostrar la memoria leyéndola como memoria de código. El editor hará las lecturas usando MOVC, que realmente emplea la señal PSEN para habilitar la lectura de la memoria. La configuración actual de lectura se muestra en la esquina superior izquierda de la pantalla. Incluso cuando nos encontramos en modo de visualización de memoria CODE, todos los intentos de escritura se realizan mediante la instrucción MOVX, porque no hay instrucción que pueda escribir la memoria de código. La edición no está deshabilitada cuando se selecciona memoria de código (CODE).
- CTRL-D – Mostrar como memoria de datos DATA (MOVX) : Fuerza al editor a mostrar la memoria mediante lectura de memoria de datos (DATA). El editor realizará todas las lectura con MOVX, que realmente usar la señal RD para habilitar la memoria. La configuración actual de lectura se muestra en la zona superior izquierda de la pantalla.
- CTRL-L – Redibujar la pantalla : Redibuja completamente la pantalla... útil con programas que tengan una emulación de terminal VT100 pobre, o PCs lentos con UART 16450 o 8250 UART donde se pierden algunas veces caracteres.
- CTRL-Q (o ESC): Salir : Salir del editor de memoria y volver a PAULMON2.

7.- Funciones incluidas que puedes usar en tus programas.

PAULMON2 dispone de varias funciones internas que se pueden acceder desde los programas. La mayoria de estas funciones proporcionan E/S para el puerto serie. Para usarlas, debemos copiar líneas del fichero PAULMON2.EQU, donde aparecen las direcciones de cada rutina. A continuación se ofrece un pequeño listado:

- Cout: Imprime el byte en acc al puerto serie
- Cin : Obtiene un carácter del puerto serie. Esta rutina esperará hasta que llegue un carácter, si no hay uno en el buffer. El carácter recibido se devuelve en Acc.
- Cin_filter : Funciona como cin, pero también intenta identificar secuencias de escape multi-byte empleados por algunas de las teclas del teclado PC. Cuando se observan algunas de estas secuencias, solo se devuelve un carácter simple. Se emplean bucle de temporización con la velocidad en baudios, de forma que la tecla ESC se detecte correctamente.

Tecla	Valor retorno	Secuencia escape
Arriba	11 (^K)	1B 5B 41
Abajo	10 (^J)	1B 5B 42
Derecha	21 (^U)	1B 5B 43
Izquierda	8 (^H)	1B 5B 44
Pág. Arriba	25 (^Y)	1B 5B 35 7E
Pág. Abajo	26 (^Z)	1B 5B 36 7E

- Phex : Imprime el valor del acumulador como un número hexadecimal de 2 dígitos
- Phex16 : Imprime el valor de DPTR como un número hexadecimal de 4-dígitos
- Pstr : Imprime una cadena, localizada en memoria de código (CODE). DPTR debe cargarse con la dirección del primer byte de la cadena. La cadena debe terminar con cero (que no se envía), o un byte con el bit más significativo a uno (que se manda con el msb puesto a cero). DPTR se devuelve con su valor apuntando al siguiente byte de memoria tras el final de la cadena.
- Phex1 : Imprime un dígito simple en HEX. Acc debe contener un valor de 0 a 15.
- Asc2hex : Convierte un carácter ascii (‘0’ a ‘9’ o ‘A’ a ‘F’) al número correspondiente que se devuelve en el acumulador. Si el valor pasado en el acumulador no es uno de los 16 caracteres legales, el bit de carry se pone a 1, en caso contrarío es puesto a cero. Esta función no aceptará letras minúsculas, por lo que debe llamarse a la rutina upper antes si deben procesarse letras minúsculas..
- Ghex : Obtiene un número hexadecimal de 2 dígitos del puerto serie, que se devuelve en el acumulador. Si el usuario pulsa ESC, C es puesto a 1 (a 0 en caso contrario) y el valor del acumulador es desconocido. Si el usuario pulsa Enter sin producir entrada, PSW.5 es puesto a 1 (a 0 en caso contrario), y el valor del acumulador es desconocido. R2 y R3 resultan modificados.
- Ghex16 : Obtiene un número hexadecimal de 4 dígitos del puerto serie, que se devuelve en el acumulador. Si el usuario pulsa ESC, C es puesto a 1 (a 0 en caso contrario) y el valor del acumulador es desconocido. Si el usuario pulsa Enter sin producir entrada, PSW.5 es puesto a 1 (a 0 en caso contrario), y el valor del acumulador es desconocido. ACC, R2, R3, R4 y R5 resultan modificados.
- Esc : Comprueba si hay un byte esperando en el puerto serie C=0 si no, C=1 si el buffer está ocupado
- Upper : Convierte el carácter en ACC a mayúsculas, si es un carácter minúscula. Si hay un carácter no alfanumérico en ACC, no resulta modificado.
- Autobaud : Realiza una detección automática de velocidad (espera que el usuario pulse Enter)
- Pcstr : Imprime una cadena en formato comprimido. El formato de compresión “al vuelo” usado en PAULMON2 no está documentado, desafortunadamente.
- Newline : Imprime un retorno de carro y nueva línea al puerto serie.
- Lenstr : Mide la longitud de una cadena, localizada en memoria CODE. DPTR debe cargarse con la dirección del primer byte de la cadena. La cadena debe terminar como se indica en pstr. DPTR es devuelto con su valor apuntando al siguiente byte en memoria al final de la cadena. La longitud de la cadena es devuelta en R0. Obviamente, las cadenas mayores de 255 caracteres no se pueden medir, pero pstr debe manejar cadenas largas sin problemas.
- Pint8u : Imprime el valor en ACC como entero sin signo (base 10). En otras palabras, se imprimirá un número entre 0 y 255. La bandera F0 resulta modificada.
- Pint8 : Imprime el valor en ACC como entero con signo (base 10). En otras palabras, se imprimirá un número entre -128 y 127. La bandera F0 resulta modificada.
- Pint16u : Imprime el valor en DPTR como un entero sin signo (base 10). En otras palabra, se imprimirá un número entre 0 y 65535. Resultan modificados los registros R2, R3, R4, R5, y la bandera PSW.5.
- Smart_wr : Intenta escribir un byte del ACC a la dirección de memoria especificada por DPTR. Si PAULMON2 está configurado para usar Flash ROM, y la dirección especificada de memoria está en la Flash ROM, **smart_wr** llamará la rutina **prgm** para que el byte resulte escrito con el algoritmo de programación de la flash. El acarreo se pone a 0 si el byte fue escrito de forma correcta, o a 1 si no pudo ser verificado.

- Prgm : Escribe un byte de ACC a la Flash ROM en la dirección DPTR. El acarreo es puesto a cero si el byte fue escrito con éxito, o puesto a 1 si no pudo verificarse. Las interrupciones se deshabilitan durante un breve periodo de tiempo.
- Erall : Borra la ROM. Esto puede llevar algún tiempo.
- Find : Busca una cabecera de programa en la memoria. Esta función desafortunadamente no esta bien documentada.

PAULMON2 contiene una tabla de instrucciones de salto localizada inmediatamente después de los vectores de interrupción. Los símbolos definidos en PAULMON2.EQU son las direcciones de estas instrucciones de salto. Deben usarse estas direcciones en lugar del lugar donde se encuentra realmente el código en PAULMON2, porque las direcciones reales de código pueden cambiar en el futuro si se actualiza PAULMON2. Las direcciones de la tabla de saltos no cambiarán, para mantener la compatibilidad con el código existente.

8.- Añadir comandos.

Se pueden añadir fácilmente comandos a PAULMON2 colocando la cabecera necesaria. Estos comandos pueden añadir nueva funcionalidad o reemplazar a los comandos existentes. El paquete de extras es un ejemplo de tres de estos comandos. El listado y paso a paso añaden nuevos comandos, y el editor de memoria reemplaza al editor integrado usando la misma tecla ("E").

La creación de comandos externos con la cabecera de programa constituye una forma muy fácil de añadir funcionalidad a PAULMON2, porque cada nueva versión puede descargarse y probarse en RAM sin reprogramar la EPROM con PAULMON2.

Aquí tenemos un ejemplo muy simple:

```
.equ    locat, 0x3D00     ;Location for this code

;This very simple paulmon2 command sets the memory location
;pointer to 3000 when the '3' key is pressed.  That's a very simple
;command, but it can be nice when developing a program for
;use at 3000 when the default memory location is at 2000...
;Just press '3' at the prompt rather than typing "N" and "3000"

.org    locat
.db     0xA5,0xE5,0xE0,0xA5     ;signature bytes
.db     254,'3',0,0             ;id (254=cmd)
.db     0,0,0,0                 ;prompt code vector
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;user defined
.db     255,255,255,255         ;length and checksum (255=unused)
.db     "Memory Location 3000",0
.org    locat+64                ;executable code begins here

        mov     r6, #0
        mov     r7, #0x30
        ret
```

El símbolo “locat" hace que el reensamblado de este comando en otra dirección de memoria sea más fácil. El grupo de directivas ".db" crea la cabecera. El código solo contiene tres instrucciones. El "puntero de memoria " que emplea PAULMON2 se guarda en los registros R6 y R7, por lo que estos dos registros se sobrescriben con el valor 3000, y finalmente el comando retorna a PAULMON2 con la instrucción "RET".

Cuando este comando simple se descarga en PAULMON2:

```
PAULMON2 (beta7) Loc:2000 > Download

Begin ascii transfer of Intel hex file, or ESC to abort

......
Download completed

Summary:
 6  lines received
 58  bytes received
 58  bytes written
No errors detected
```

Inmediatamente después de que el programa se coloca en memoria, está disponible para usarse en PAULMON2. El comando de listado de programas ("M") debe mostrar este programa en la lista si la cabecera es correcta:

```
PAULMON2 (beta7) Loc:2000 > List programs

Program Name                      Location      Type
  List                              1000      External command
  Single-Step                       1400      External command
  Memory Editor (VT100)             1800      External command
  Memory Location 3000              3D00      External command
```

Y el comando de ayuda debe encontrar la cabecera para este nuevo programa y mostrar correctamente su nombre y la tecla que ejecuta el comando:

```
PAULMON2 (beta7) Loc:2000 > Help

Standard Commands
  ?- This help list
  M- List programs
  R- Run program
  D- Download
  U- Upload
  N- New location
  J- Jump to memory location
  H- Hex dump external memory
  I- Hex dump internal memory
  E- Editing external ram
  C- Clear memory
  Z- Erase flash rom
User Installed Commands
  L- List
  S- Single-Step
  E- Memory Editor (VT100)
  3- Memory Location 3000
```

Finalmente, cuando el programa se ejecuta, no esperamos que nada dramatico suceda, excepto el cambio en el puntero de dirección de datos de PAULMON2. Esto es lo que realmente sucede cuando pulsamos "3":

```
PAULMON2 (beta7) Loc:2000 >  Memory Location 3000
PAULMON2 (beta7) Loc:3000 >
```

Como resumen, estas son las reglas para escribir comandos de PAULMON2:

- Usar una cabecera de programa de 64-bytes, tipo 254.

- R6 y R7 guardan el puntero de memoria de PAULMON2. Los otros 6 registros, ACC, B pueden cambiar. R6 y R7 pueden modificarse, pero esto tendrá reflejo en el valor del puntero de memoria cuando el código retorne a PAULMON2.

- No modificar el puntero de pila. Si el valor del puntero de pila es diferente o los valores que PAULMON2 introduce en la pila antes de ejecutar el comando se alteran, PAULMON2 probablemente fallara cuando retorne el comando.

- No modificar timer1 o la configuración del puerto serie (o restaurar con cuidado si lo hacemos). La UART integrada es necesaria cuando el código retorna a PAULMON2. Se asume que nuestro comando externo no modifica los registros de función especial.

- Es aconsejable realizar todos las E/S serie empleando las rutinas de E/S de PAULMON2. Ver PAULMON2.EQU para los detalles. Si se necesita almacenamiento de datos estático entre llamadas al nuevo comando, se debe seleccionar memoria que no entre en conflicto con la pila, los 8 registros, los 16 bytes de espacio temporal usado por el comando de descarga, y el parámetro de velocidad en baudios. (añadir enlaces que marquen donde residen todas estas variables).

- El nuevo comando debe ser capaz de funcionar si cualquier variable estática es sobrescrita, porque PAULMON2 puede usarse para saltar a un programa que pueda sobrescribir cualquier zona de memoria mientras tiene el control del 8051.

- Cualquier comando de PAULMON2 puede reemplazarse con un comando externo. No es mala idea reemplazar el comando de descarga ("D"), y quizás otros también.

- Si nuestro nuevo comando hace uso de interrupciones (el comando paso a paso es un ejemplo de esto), no retornar a PAULMON2 mientras nos encontramos dentro de una rutina de servicio de interrupción. Si nuestro comando debe cancelarse mientras está sirviendo una interrupción, LJMP a la dirección cero, donde PAULMON2 limpiará cualquier interrupción pendiente con dos instrucciones RETI.

- Volver a PAULMON2 con una instrucción "RET".

Comandos de Ejemplo

Estos ejemplos pueden resultar útiles como punto de partida para crear nuestros propios comandos. Son muy similares al comando de volcado de memoria, pero cada uno muestra el doble de datos.

```
.equ  locat, 0x3E00          ;Location for this code

;This file demonstrates two external commands which are
;nearly identical to the built-in ones inside PAULMON2.
;The external memory dump prints 512 bytes (32 lines)
;and the internal memory dump prints all 256 locations.

.equ  dump_key, 'H'          ;hex dump memory
.equ  intm_key, 'I'          ;hex dump internal memory

.equ  cout, 0x0030           ;routines to use from paulmon2
.equ  phex, 0x0034
.equ  phex16, 0x0036
.equ  newline, 0x0048

.org  locat
.db   0xA5,0xE5,0xE0,0xA5    ;signature bytes
.db   254,dump_key,0,0       ;id (254=cmd)
.db   0,0,0,0                ;prompt code vector
.db   0,0,0,0                ;reserved
.db   0,0,0,0                ;reserved
```

```
.db   0,0,0,0                 ;reserved
.db   0,0,0,0                 ;user defined
.db   255,255,255,255         ;length and checksum (255=unused)
.db   "Hex Dump External Memory",0
.org  locat+64                ;executable code begins here

dump:
            mov   r2, #32         ;number of lines to print
            lcall newline
            lcall newline
dump1:      mov   dpl, r6
            mov   dph, r7
            lcall phex16          ;tell 'em the memory location
            mov   a,#':'
            lcall cout
            acall space
            mov   r3, #16         ;r3 counts # of bytes to print
            mov   dpl, r6
            mov   dph, r7
dump2:      clr   a
            movc  a, @a+dptr
            inc   dptr
            lcall phex            ;print each byte in hex
            acall space
            djnz  r3, dump2
            acall space           ;print a couple extra space
            acall space
            mov   r3, #16
            mov   dpl, r6
            mov   dph, r7
dump3:      clr   a
            movc  a, @a+dptr
            inc   dptr
            anl   a, #01111111b   ;avoid unprintable characters
            cjne  a, #127, dump3b
            clr   a           ;avoid 127/255 (delete/rubout) char
dump3b:     add   a, #224
            jc    dump4
            clr   a           ;avoid control characters
dump4:      add   a, #32
            lcall cout
            djnz  r3, dump3
            lcall newline
            mov   r6, dpl
            mov   r7, dph
            djnz  r2, dump1       ;loop back up to print next line
            lcall newline
            ret

space:      mov   a, #' '
            lcall cout
            ret


.org    locat+256
.db     0xA5,0xE5,0xE0,0xA5     ;signature bytes
.db     254,intm_key,0,0        ;id (254=cmd)
.db     0,0,0,0                 ;prompt code vector
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;reserved
```

```
.db     0,0,0,0                 ;user defined
.db     255,255,255,255         ;length and checksum (255=unused)
.db     "Hex Dump Internal Memory",0
.org    locat+256+64            ;executable code begins here

intm:       lcall newline
            lcall newline
            mov   r0, #0
            sjmp  intm3
intm2:      lcall newline
            cjne  r0, #0, intm3
            ljmp  newline
intm3:      mov   a, r0
            lcall phex
            mov   a, #':'
            lcall cout
intm4:      acall space
            mov   a, @r0
            lcall phex
            inc   r0
            mov   a, r0
            anl   a, #00001111b
            jnz   intm4
            sjmp  intm2
```

El procedimiento para usar estas es el mismo que el indicado anteriormente. Solo descargar el fichero y una vez en memoria, pulsar "H" o "I" ejecutará el código mostrado aquí en lugar del volcado original de memoria integrados en PAULMON2.

Aunque estos son ejemplo sencillos, se pueden hacer bastantes cosas con comandos externos... de hecho casi cualquier cosa que ha sido incluida en PAULMON2 puede añadirse como comando externo. El paquete de extras es un ejemplo de comando mucho más complejos.

For boards with Flash ROM or other unused non-volatile memory, external commands can be used to make permanent additions to PAULMON2 which are nearly as easy to develop as ordinary applications, because PAULMON2 itself does not have to be modified to add the functionality.

9.- Problemas más comunes.

Estos son los problemas más frecuentes:

- El programador de EPROM para programar PAULMON2 ha leído el fichero como binario en lugar de intel-hex y la EPROM contiene un código incorrecto.

- Las líneas del puerto serie (2 y 3) están cambiadas y no aparece el mensaje de bienvenida.

- La detección automática de velocidad del puerto serie capta un ruido o carácter erróneo mientras la placa está conectada y arrancamos el programa emulador de terminal.